

NNN	NNN	IIIIIIIIII	CCCCCCCCCCCC	NNN	NNN	FFFFFFFFFFFFFF
NNN	NNN	IIIIIIIIII	CCCCCCCCCCCC	NNN	NNN	FFFFFFFFFFFFFF
NNN	NNN	IIIIIIIIII	CCCCCCCCCCCC	NNN	NNN	FFFFFFFFFFFFFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNNNNN	NNN	III	CCC	NNNNNN	NNN	FFF
NNNNNN	NNN	III	CCC	NNNNNN	NNN	FFF
NNNNNN	NNN	III	CCC	NNNNNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFFFFFFFFFFFFF
NNN	NNN	III	CCC	NNN	NNN	FFFFFFFFFFFFFF
NNN	NNN	III	CCC	NNN	NNN	FFFFFFFFFFFFFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	III	CCC	NNN	NNN	FFF
NNN	NNN	IIIIIIIIII	CCCCCCCCCCCC	NNN	NNN	FFF
NNN	NNN	IIIIIIIIII	CCCCCCCCCCCC	NNN	NNN	FFF
NNN	NNN	IIIIIIIIII	CCCCCCCCCCCC	NNN	NNN	FFF

```
CCCCCCCC  NN      NN  FFFFFFFF  MM      MM  AAAAAA  IIIIII  NN      NN
CCCCCCCC  NN      NN  FFFFFFFF  MM      MM  AAAAAA  IIIIII  NN      NN
CC         NN      NN  FF          MMMM  MMMM  AA      AA  II      II  NN      NN
CC         NN      NN  FF          MMMM  MMMM  AA      AA  II      II  NN      NN
CC         NNNN     NN  FF          MM  MM  AA      AA  II      II  NNNN     NN
CC         NNNN     NN  FF          MM  MM  AA      AA  II      II  NNNN     NN
CC         NN  NN  NN  FFFFFFFF  MM      MM  AA      AA  II      II  NN  NN  NN
CC         NN  NN  NN  FFFFFFFF  MM      MM  AA      AA  II      II  NN  NN  NN
CC         NN      NNNN  FF          MM      MM  AAAAAAAAAA  II      II  NN      NNNN
CC         NN      NNNN  FF          MM      MM  AAAAAAAAAA  II      II  NN      NNNN
CC         NN      NN  FF          MM      MM  AA      AA  II      II  NN      NN
CC         NN      NN  FF          MM      MM  AA      AA  II      II  NN      NN
CC         NN      NN  FF          MM      MM  AA      AA  IIIIII  NN      NN
CCCCCCCC  NN      NN  FF          MM      MM  AA      AA  IIIIII  NN      NN
CCCCCCCC  NN      NN  FF          MM      MM  AA      AA  IIIIII  NN      NN
```

```
LL         IIIIII  SSSSSSSS
LL         IIIIII  SSSSSSSS
LL         II      SS
LL         II      SS
LL         II      SS
LL         II      SS
LL         II      SSSSSS
LL         II      SSSSSS
LL         II      SS
LL         II      SS
LL         II      SS
LL         II      SS
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS
```

```
0001 0 XTITLE 'DECnet Ethernet Configurator Module'
0002 0 MODULE CNFMAIN (
0003 0     LANGUAGE (BLISS32),
0004 0     IDENT = 'V04-000',
0005 0     MAIN = CNFSMAIN
0006 0 ) =
0007 1 BEGIN
0008 1
0009 1
0010 1 *****
0011 1 *
0012 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0013 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0014 1 *  ALL RIGHTS RESERVED.
0015 1 *
0016 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0017 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0018 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0019 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0020 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0021 1 *  TRANSFERRED.
0022 1 *
0023 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0024 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0025 1 *  CORPORATION.
0026 1 *
0027 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0028 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0029 1 *
0030 1 *****
0031 1
0032 1
0033 1 ++
0034 1 FACILITY:    DECnet Configurator Module (NICONFIG)
0035 1
0036 1 ABSTRACT:
0037 1
0038 1     This module contains the main entry for NICONFIG, which
0039 1     provides the DECnet Configurator Module, as well as a
0040 1     few routines of general utility.
0041 1
0042 1     NICONFIG listens to the system ID messages broadcast
0043 1     regularly by devices on the NI and maintains a data
0044 1     base which can be queried.
0045 1
0046 1     To issue commands to NICONFIG, the user uses NCP, which
0047 1     generates messages in the NICE protocol which it passes to NML.
0048 1     NICONFIG is started by the network in response to a
0049 1     request for a logical link connection by NML. NML then
0050 1     passes the NICE message, in tact, to NICONFIG for processing.
0051 1
0052 1 ENVIRONMENT: VAX/VMS Operating System
0053 1
0054 1     NICONFIG requires the following privileges for proper execution:
0055 1     LOG_IO, SYSNAM
0056 1
0057 1
```

CNFMAIN
V04-000

DECnet Ethernet Configurator Module

G 11
16-Sep-1984 02:02:49
14-Sep-1984 12:49:51

VAX-11 Bliss-32 V4.0-742
[NICNF.SRC]CNFMAIN.B32;1

Page 2
(1)

..	58	0058	1	AUTHOR:	Bob Grosso,	CREATION DATE:	13-Oct-1982
..	59	0059	1				
..	60	0060	1	MODIFIED BY:			
..	61	0061	1				
..	62	0062	1				
..	63	0063	1				
..	64	0064	1				
..	65	0065	1				
..	66	0066	1				
..	67	0067	1				
..	68	0068	1				
..	69	0069	1				
..	70	0070	1				
..	71	0071	1	--			

	V03-003	RPG0003	Bob Grosso	16-May-1983
			Correct zero virtual memory bug.	
	V03-002	RPG0002	Bob Grosso	02-May-1983
			Check for NETMBX and TMPMBX privileges.	
	V03-001	RPG0001	Bob Grosso	10-Mar-1983
			Look for require file in SRC\$ directory.	

```
73 0072 1 XSBTTL 'Definitions'
74 0073 1
75 0074 1
76 0075 1 INCLUDE FILES:
77 0076 1
78 0077 1
79 0078 1 LIBRARY 'SYSS$LIBRARY:STARLET';      ! VMS common definitions
80 0079 1
81 0080 1 LIBRARY 'SHRLIB$:NET';              ! Network definitions
82 0081 1
83 0082 1 REQUIRE 'LIB$:CNFDEF.R32';
84 0173 1
85 0174 1 REQUIRE 'SRC$:CNFPREFIX.REQ';        ! Collection of useful macros
86 0271 1                                     ! and literals
87 0272 1
88 0273 1 BUILTIN functions
89 0274 1
90 0275 1
91 0276 1 BUILTIN
92 0277 1     INSQUE,                          ! INSQUE instruction
93 0278 1     REMQUE;                          ! REMQUE instruction
94 0279 1
95 0280 1
96 0281 1 LITERALS
97 0282 1
98 0283 1
99 0284 1 GLOBAL LITERAL
100 0285 1
101 0286 1     CNF$C_MAXMBXMSG = 124,            ! Maximum size of mailbox message
102 0287 1     CNF$C_SYNCH_EFN = 1,             ! Synchronous event flag number
103 0288 1     CNF$C_ASYNC_EFN = 2,            ! Asynchronous event flag number
104 0289 1     CNF$C_STARTUP_EFN = 3;          ! Event flag number for startup timer
105 0290 1
106 0291 1
107 0292 1
108 0293 1 OWN STORAGE:
109 0294 1
110 0295 1
111 0296 1 GLOBAL
112 0297 1     CNF$GL_LOGMASK : BITVECTOR [32], ! Logging control mask
113 0298 1
114 0299 1     CNF$GQ_CIRSURLST : VECTOR [2],      ! List of circuit under surveillance
115 0300 1     CNF$GQ_IRBLST : VECTOR [2],      ! Listhead for incoming links
116 0301 1     CNF$A_MBXMSG : VECTOR [CNF$C_MAXMBXMSG, BYTE], ! Mailbox message buffer
117 0302 1
118 0303 1     CNF$W_NETCHAN : WORD,                ! Channel opened to network
119 0304 1     CNF$W_MBXCHAN : WORD,             ! Channel to mailbox
120 0305 1     CNF$B_SURVEILLANCE_SET,          ! Boolean: mark if surveillance has been set
121 0306 1     CNF$B_STARTING_UP;                ! Boolean: mark if still starting up
122 0307 1
123 0308 1 OWN
124 0309 1     CNF$Q_A_STARTUP_WAIT :                ! ASCII wait delta time (3 minutes)
125 0310 1     BBLOCK [DSC$C_S_BLN]
126 0311 1     INITIAL (XCHARCOUNT ('0 00:03:00.00'),
127 0312 1     UPLIT PSECT ($OWN$) (XASCII '0 00:03:00.CO')),
128 0313 1
129 0314 1     CNF$Q_B_STARTUP_WAIT : VECTOR [2, LONG], ! Time in binary converted from ASCII
```

```
130 0315 1 CNF$$_VM; ! Tally of virtual memory allocated
131 0316 1
132 0317 1
133 0318 1 ! TABLE OF CONTENTS:
134 0319 1 !
135 0320 1
136 0321 1 FORWARD ROUTINE
137 0322 1
138 0323 1 CNF$$_MAIN, ! Main entry
139 0324 1 CHECK_PRIVS : NOVALUE, ! Check that NICONFIG is executing with sufficient privileges
140 0325 1 INIT_LOG : NOVALUE, ! Initialize for debug logging
141 0326 1 INIT_DATA : NOVALUE, ! Initialize data structures
142 0327 1 DECLARE_OBJNAM : NOVALUE, ! Declare $NICONFIG to the Net
143 0328 1 SET_TIME_BOMB : NOVALUE, ! Set timer to verify a valid SET command was received
144 0329 1 TIME_BOMB : NOVALUE, ! Queue work item to abort if there are no surveillance requests
145 0330 1 TERMINATE_GRACE : NOVALUE, ! Terminate the grace period
146 0331 1 CNF$$_TRACE : NOVALUE, ! Log messages to log file
147 0332 1 CNF$$_LOG_DATA : NOVALUE, ! Log messages to log file
148 0333 1 CNF$$_EXIT : NOVALUE, ! Clean up and exit
149 0334 1
150 0335 1
151 0336 1 !
152 0337 1 ! EXTERNAL REFERENCES:
153 0338 1 !
154 0339 1
155 0340 1 EXTERNAL ROUTINE
156 0341 1
157 0342 1 ! Module CNFINTRPT
158 0343 1
159 0344 1 CNF$$_SOLICIT_INTERRUPT : NOVALUE, ! Solicit work items
160 0345 1
161 0346 1 ! Module CNFWORKQ
162 0347 1
163 0348 1 WKQ$ADD_WORK_ITEM, ! Add work to the work queue
164 0349 1 WKQ$DO_WORK_ITEM; ! Perform work on work queue
165 0350 1
166 0351 1 EXTERNAL ROUTINE
167 0352 1
168 0353 1 LIB$ASN_WTH_MBX : ADDRESSING_MODE (GENERAL),
169 0354 1 LIB$CVT_HTB : ADDRESSING_MODE (GENERAL),
170 0355 1 LIB$GET_VM : ADDRESSING_MODE (GENERAL),
171 0356 1 LIB$FREE_VM : ADDRESSING_MODE (GENERAL),
172 0357 1 LIB$PUT_OUTPUT : ADDRESSING_MODE (GENERAL);
173 0358 1
174 0359 1 EXTERNAL LITERAL
175 0360 1
176 0361 1 CNF$$_GETVM, ! Allocated !UL bytes of virtual memory, total of !UL
177 0362 1 CNF$$_FAILFREEVM, ! Failed to deallocate !UL bytes of virtual memory
178 0363 1 CNF$$_FAILGETVM, ! Failed to allocate !UL bytes of virtual memory
179 0364 1 CNF$$_FREEVM, ! Deallocated !UL bytes of virtual memory leaving !UL
180 0365 1 CNF$$_LOGIC, ! Program logic error, or unexpected condition
181 0366 1 CNF$$_LOGIO, ! NICONFIG requires LOG_IO privilege
182 0367 1 CNF$$_NETASN, ! Failed to declare name to network
183 0368 1 CNF$$_NETMBX, ! NICONFIG requires NETMBX privilege
184 0369 1 CNF$$_PRIV, ! Privilege error
185 0370 1 CNF$$_SYSNAM, ! NICONFIG requires SYSNAM privilege
186 0371 1 CNF$$_TMPMBX; ! NICONFIG requires TMPMBX privilege
```

CNFMAIN
V04-000

DECnet Ethernet Configurator Module
Definitions

; 187

0372 1

J 11
16-Sep-1984 02:02:49
14-Sep-1984 12:49:51

VAX-11 Bliss-32 V4.0-742
[NICNF.SRC]CNFMAIN.B32;1

Page 5
(2)

CN
VO

6

```
189 0373 1 XSBTTL 'CNFSMAIN Main Entry'
190 0374 1 GLOBAL ROUTINE CNFSMAIN =
191 0375 1
192 0376 1
193 0377 1 ++
194 0378 1 FUNCTIONAL DESCRIPTION:
195 0379 1 This is the main entry point for the Configurator Module.
196 0380 1 It calls the initialization routines and sits in a loop
197 0381 1 performing work from the work queue.
198 0382 1 If after the termination of the startup grace period,
199 0383 1 no work requests have specified that NICONFIG place one
200 0384 1 or more circuits under surveillance, it will quietly go
201 0385 1 away.
202 0386 1
203 0387 1 FORMAL PARAMETERS:
204 0388 1 NONE
205 0389 1
206 0390 1 IMPLICIT INPUTS:
207 0391 1 NONE
208 0392 1
209 0393 1 IMPLICIT OUTPUTS:
210 0394 1 NONE
211 0395 1
212 0396 1 ROUTINE VALUE:
213 0397 1 COMPLETION CODES:
214 0398 1 NONE
215 0399 1
216 0400 1 SIDE EFFECTS:
217 0401 1 NONE
218 0402 1
219 0403 1 --
220 0404 1
221 0405 2 BEGIN
222 0406 2
223 0407 2 CHECK_PRIVS (); ! Ensure that NICONFIG is executing with sufficient privilege
224 0408 2
225 0409 2 INIT_LOG (); ! Initialize for debug logging
226 0410 2
227 0411 2 INIT_DATA (); ! Initialize data structures
228 0412 2
229 0413 2 DECLARE OBJNAM ();
230 0414 2 CNF$TRACE (DBG$C_TRACE, $DESCRIPTOR('TRACE'), $DESCRIPTOR('Object name declared'));
231 0415 2
232 0416 2
233 0417 2 | Issue a timer AST to wake up some in the future so that a check
234 0418 2 | a check can be performed to ensure that useful work is being done,
235 0419 2 | and a decision made whether or not to terminate.
236 0420 2
237 0421 2 SET_TIME_BOMB ();
238 0422 2
239 0423 2 CNF$SOLICIT_INTERRUPT (); ! See if anyone wants to issue a Net connect
240 0424 2
241 0425 2
242 0426 2 | So long as at least one circuit is under surveillance
243 0427 2 | or the startup grace period is in effect,
244 0428 2 | process the work queue.
245 0429 2
```

CNFMMAIN
V04-000

DECnet Ethernet Configurator Module
CNFSMAIN Main Entry

L 11
16-Sep-1984 02:02:49
14-Sep-1984 12:49:51

VAX-11 Bliss-32 V4.0-742
[NICNF.SRC]CNFMMAIN.B32;1

Page 7
(3)

```

: 246      0430 2      WHILE (.CNFSB_SURVEILLANCE_SET OR .CNFSB_STARTING_UP) DO
: 247      0431      BEGIN
: 248      0432          $HIBER;          ! ZZZzzZZZzzz until some work comes in
: 249      0433          CNF$TRACE (DBG$C_TRACE, $DESCRIPTOR('TRACE'), $DESCRIPTOR ('Wakeup to perform work items'));
: 250      0434
: 251      0435          WHILE WKQ$DO_WORK_ITEM () DO      ! Perform work until queue is empty
: 252      0436          END;
: 253      0437
: 254      0438          CNF$TRACE (DBG$C_TRACE, $DESCRIPTOR ('TRACE'),
: 255      0439              $DESCRIPTOR ('Aborting --- No surveillance requested'));
: 256      0440
: 257      0441          CNF$EXIT (SS$ NORMAL);      ! Exit sucessfully
: 258      0442          RETURN SS$_NORMAL;      ! Added for completeness
: 259      0443          END;      ! MAIN routine
```

```

                                .TITLE CNFMMAIN DECnet Ethernet Configurator Module
                                .IDENT  \V04-000\
                                .PSECT $SPLITS$,NOWRT,NOEXE,2

                                45 43 41 52 54 00000 P.AAC: .ASCII \TRACE\
                                00005          .BLKB 3
                                00000005 00008 P.AAB: .LONG 5
                                00000000' 0000C          .ADDRESS P.AAC
63 65 64 20 65 6D 61 6E 20 74 63 65 6A 62 4F 00010 P.AAE: .ASCII \Object name declared\
64 65 72 61 6C 0001F
                                00000014 00024 P.AAD: .LONG 20
                                00000000' 00028          .ADDRESS P.AAE
                                45 43 41 52 54 0002C P.AAG: .ASCII \TRACE\
                                00031          .BLKB 3
                                00000005 00034 P.AAF: .LONG 5
                                00000000' 00038          .ADDRESS P.AAG
6F 66 72 65 70 20 6F 74 20 70 75 65 6B 61 57 0003C P.AAI: .ASCII \Wakeup to perform work items\
73 6D 65 74 69 20 6B 72 6F 77 20 6D 72 0004B
                                0000001C 00058 P.AAH: .LONG 28
                                00000000' 0005C          .ADDRESS P.AAI
                                45 43 41 52 54 00060 P.AAK: .ASCII \TRACE\
                                00065          .BLKB 3
                                00000005 00068 P.AAJ: .LONG 5
                                00000000' 0006C          .ADDRESS P.AAK
6F 4E 20 2D 2D 2D 2D 67 6E 69 74 72 6F 62 41 00070 P.AAM: .ASCII \Aborting --- No surveillance requested\
72 20 65 63 6E 61 6C 6C 69 65 76 72 75 73 20 0007F
64 65 74 73 65 75 71 65 0008E
                                00096          .BLKB 2
                                00000026 00098 P.AAL: .LONG 38
                                00000000' 0009C          .ADDRESS P.AAM

                                .PSECT $OWNS$,NOEXE,2

00 00 30 30 2E 30 30 3A 33 30 3A 30 30 20 30 00000 P.AAA: .ASCII \0 00:03:00.00\<0><0><0>
                                00 0000F
                                0000000D 00010 CNF$Q_A_STARTUP_WAIT:
                                .LONG 13
                                00000000' 00014          .ADDRESS P.AAA
                                00018 CNF$Q_B_STARTUP_WAIT:
                                .BLKB 8
```

00020 CNF\$\$_VM:
 .BKLB 4
 .PSECT \$GLOBAL\$,NOEXE,2

00000 CNF\$GL_LOGMASK::
 .BKLB 4
00004 CNF\$GQ_CIRSURLST::
 .BKLB 8
0000C CNF\$GQ_IRBLST::
 .BKLB 8
00014 CNF\$A_MBXMSG::
 .BKLB 124
00090 CNF\$W_NETCHAN::
 .BKLB 2
00092 CNF\$W_MBXCHAN::
 .BKLB 2
00094 CNF\$B_SURVEILLANCE_SET::
 .BKLB 4
00098 CNF\$B_STARTING_UP::
 .BKLB 4

CNF\$C_MAXMBXMSG== 124
CNF\$C_SYNCH_EFN== 1
CNF\$C_ASYNC_EFN== 2
CNF\$C_STARTUP_EFN== 3

.EXTRN CNF\$SOLICIT_INTERRUPT
.EXTRN WKQ\$ADD_WORK_ITEM
.EXTRN WKQ\$DO_WORK_ITEM
.EXTRN LIB\$ASN_WTH_MBX
.EXTRN LIB\$CVT_HTB, LIB\$GET_VM
.EXTRN LIB\$FREE_VM, LIB\$PUT_OUTPUT
.EXTRN CNF\$GETVM, CNF\$FAILFREEVM
.EXTRN CNF\$FAILGETVM, CNF\$FREEVM
.EXTRN CNF\$LOGIC, CNF\$LOGIO
.EXTRN CNF\$NETASN, CNF\$NETMBX
.EXTRN CNF\$PRIV, CNF\$SYSNAM
.EXTRN CNF\$TMPMBX, SYS\$HIBER

.PSECT \$CODE\$,NOWRT,2

			000C	00000	.ENTRY	CNF\$MAIN, Save R2,R3	: 0374
	53	0000V	CF	9E 00002	MOVAB	CNF\$TRACE, R3	
	52	0000'	CF	9E 00007	MOVAB	P.AAD, R2	
0000V	CF		00	FB 0000C	CALLS	#0, CHECK_PRIVS	: 0407
0000V	CF		00	FB 00011	CALLS	#0, INIT_LOG	: 0409
0000V	CF		00	FB 00016	CALLS	#0, INIT_DATA	: 0411
0000V	CF		00	FB 0001B	CALLS	#0, DECLARE_OBJNAM	: 0413
			52	DD 00020	PUSHL	R2	: 0414
		E4	A2	9F 00022	PUSHAB	P.AAB	
			01	DD 00025	PUSHL	#1	
	63		03	FB 00027	CALLS	#3, CNF\$TRACE	
0000V	CF		00	FB 0002A	CALLS	#0, SET TIME BOMB	: 0421
0000G	CF		00	FB 0002F	CALLS	#0, CNF\$SOLICIT_INTERRUPT	: 0423
	05	0000'	CF	E8 00034 1\$:	BLBS	CNF\$B_SURVEILLANCE_SET, 2\$: 0430
	1C	0000'	CF	E9 00039	BLBC	CNF\$B_STARTING_UP, -4\$	
00000000G	00		00	FB 0003E 2\$:	CALLS	#0, SYS\$HIBER	: 0431

CNFMAIN
V04-000

DECnet Ethernet Configurator Module
CNFSMAIN Main Entry

N 11
16-Sep-1984 02:02:49
14-Sep-1984 12:49:51

VAX-11 Bliss-32 V4.0-742
[NICNF.SRC]CNFMAIN.B32;1

Page 9
(3)

		34	A2	9F	00045	PUSHAB	P.AAH	0433
		10	A2	9F	00048	PUSHAB	P.AAF	
			01	DD	0004B	PUSHL	#1	
	63		03	FB	0004D	CALLS	#3, CNFSTRACE	
0000G	CF		00	FB	00050	CALLS	#0, WKQSDO_WORK_ITEM	0435
	DC		50	E9	00055	BLBC	R0, 1\$	
			F6	11	00058	BRB	3\$	
		74	A2	9F	0005A	PUSHAB	P.AAL	0439
		44	A2	9F	0005D	PUSHAB	P.AAJ	0438
			01	DD	00060	PUSHL	#1	
	63		03	FB	00062	CALLS	#3, CNFSTRACE	
			01	DD	00065	PUSHL	#1	0441
0000V	CF		01	FB	00067	CALLS	#1, CNFSEXIT	
	50		01	D0	0006C	MOVL	#1, R0	0442
			04	0006F	RET			0443

; Routine Size: 112 bytes, Routine Base: \$CODE\$ + 0000

```
261 0444 1 %SBTTL 'check_privs Check execution privileges'
262 0445 1 ROUTINE CHECK_PRIVS : NOVALUE =
263 0446 1
264 0447 1 ++
265 0448 1
266 0449 1 This routine verifies that NICONFIG is executing with the proper
267 0450 1 privileges.
268 0451 1
269 0452 1 Signal those privileges which are lacking.
270 0453 1
271 0454 1 --
272 0455 1 BEGIN
273 0456 1 LOCAL
274 0457 1 ABORT,
275 0458 1 PRIVMASK : BBLOCK [8],
276 0459 1 STATUS;
277 0460 1
278 0461 1 CH$FILL (0, 8, PRIVMASK); ! Initialize to zero
279 0462 1 $SETPRV (PRVPRV = PRIVMASK); ! Obtain privileges set in CURPRV
280 0463 1
281 0464 1 ABORT = FALSE;
282 0465 1
283 0466 1 !
284 0467 1 ! Check for the required privileges
285 0468 1 !
286 0469 1 IF (NOT .PRIVMASK [PRVSV_LOG_10] OR
287 0470 1 NOT .PRIVMASK [PRVSV_SYSNAM] OR
288 0471 1 NOT .PRIVMASK [PRVSV_NETMBX] OR
289 0472 1 NOT .PRIVMASK [PRVSV_TMPMBX])
290 0473 1 THEN
291 0474 1 BEGIN
292 0475 1 SIGNAL (CNFS_PRIV);
293 0476 1 ABORT = TRUE;
294 0477 1 END;
295 0478 1
296 0479 1 IF NOT .PRIVMASK [PRVSV_LOG_10] ! For reading system ID messages
297 0480 1 THEN SIGNAL (CNFS_LOG10);
298 0481 1 IF NOT .PRIVMASK [PRVSV_SYSNAM] ! For declaring itself as a known object
299 0482 1 THEN SIGNAL (CNFS_SYSNAM);
300 0483 1 IF NOT .PRIVMASK [PRVSV_NETMBX] ! For declaring itself as a known object
301 0484 1 THEN SIGNAL (CNFS_NETMBX);
302 0485 1 IF NOT .PRIVMASK [PRVSV_TMPMBX] ! For declaring itself as a known object
303 0486 1 THEN SIGNAL (CNFS_TMPMBX);
304 0487 1
305 0488 1 IF .ABORT THEN CNF$EXIT (SS$_NORMAL); ! No point in continuing
306 0489 1 RETURN;
307 0490 1 END; ! Routine Check_privs
```

.EXTRN SYS\$SETPRV

007C 00000 CHECK_PRIVS:

				WORD	Save R2,R3,R4,R5,R6	: 0445
				MOVAB	LIB\$SIGNAL, R6	
08	00	56 00000000G	00 9E 00002	SUBL2	#8, SP	
		5E	08 C2 00009	MOVCS	#0, (SP), #0, #8, PRIVMASK	: 0461
		6E	00 2C 0000C			

Address	Instruction	Comment	Address
00000000G 00	6E DD 00011	PUSHL SP	0462
	5E DD 00012	CLRQ -(SP)	
	7E 7C 00014	CLRL -(SP)	
	7E D4 00016	CALLS #4, SYS\$SETPRV	
	04 FB 00018	CLRL ABORT	0464
	52 D4 0001F	TSTB PRIVMASK	0469
	6E 95 00021	BGEQ 1\$	
	0E 18 00023	BBC #2, PRIVMASK, 1\$	0470
0A 05 02 6E	02 E1 00025	BBC #4, PRIVMASK+2, 1\$	0471
	04 E1 00029	TSTB PRIVMASK+1	0472
	AE 95 0002E	BLSS 2\$	
	0C 19 00031	PUSHL #CNFS PRIV	0475
	8F DD 00033 1\$:	CALLS #1, LIB\$SIGNAL	
	01 FB 00039	MOVL #1, ABORT	0476
	01 D0 0003C	TSTB PRIVMASK	0479
	6E 95 0003F 2\$:	BLSS 3\$	
	09 19 00041	PUSHL #CNFS LOGIO	0480
	8F DD 00043	CALLS #1, LIB\$SIGNAL	
	01 FB 00049	BBS #2, PRIVMASK, 4\$	0481
09 6E	02 E0 0004C 3\$:	PUSHL #CNFS SYSNAM	0482
	8F DD 00050	CALLS #1, LIB\$SIGNAL	
	01 FB 00056	BBS #4, PRIVMASK+2, 5\$	0483
09 02 AE	04 E0 00059 4\$:	PUSHL #CNFS NETMBX	0484
	8F DD 0005E	CALLS #1, LIB\$SIGNAL	
	01 FB 0C064	TSTB PRIVMASK+1	0485
	AE 95 00067 5\$:	BLSS 6\$	
	09 19 0006A	PUSHL #CNFS TMPMBX	0486
	8F DD 0006C	CALLS #1, LIB\$SIGNAL	
	01 FB 00072	BLBC ABORT, 7\$	0488
	52 E9 00075 6\$:	PUSHL #1	
	01 DD 00078	CALLS #1, CNF\$EXIT	
0000V CF	01 FB 0007A	RET	0490
	04 0007F 7\$:		

; Routine Size: 128 bytes, Routine Base: \$CODES + 0070

```
0491 1 %SBTTL 'init_log      Initialize debug logging'
0492 1 ROUTINE INIT_LOG : NOVALUE =
0493
0494 1 ++
0495 1
0496 1 This routine initializes the internal logging flags for NICONFIG
0497 1 debugging. The logical name NICONFIG$LOG is translated to obtain
0498 1 a hex number which is converted to a bit mask used to control
0499 1 the type of information to be logged.
0500 1
0501 1 IMPLICIT INPUTS:
0502 1
0503 1     NICONFIG$LOG logical name
0504 1
0505 1 IMPLICIT OUTPUTS:
0506 1
0507 1     Fill in CNF$GL_LOGMASK
0508 1
0509 1 --
0510 1
0511 1 BEGIN
0512 1
0513 1 LITERAL
0514 1     RSLSIZE = 10                                ! Size of the result buffer
0515 1 ;
0516 1
0517 1 LOCAL
0518 1     RSLBFR : VECTOR [RSLSIZE, BYTE],            ! Buffer for the translation
0519 1     RSLDSC : VECTOR [2]                          ! Descriptor for the buffer
0520 1 ;
0521 1
0522 1 CNF$GL_LOGMASK = 0;                             ! Initialize the logging mask
0523 1 RSLDSC [0] = RSLSIZE;                           ! Setup the descriptor
0524 1 RSLDSC [1] = RSLBFR;
0525 1
0526 1 IF                                              ! We must get a translation
0527 1 (
0528 1     $TRNLOG                                     ! Translate the name once
0529 1     (
0530 1         LOGNAM = %ASCID 'NICONFIG$LOG',          ! This is the logical name
0531 1         RSLLEN = RSLDSC [0],                    ! Place the length here
0532 1         RSLBUF = RSLDSC [1],                    ! Place the translation here
0533 1     )
0534 1 )
0535 1
0536 1 EQL
0537 1 SSS_NORMAL
0538 1 THEN
0539 1     LIB$CVT_HTB
0540 1     (
0541 1         .RSLDSC [0],
0542 1         .RSLDSC [1],
0543 1         CNF$GL_LOGMASK
0544 1     );
0545 1
0546 1 RETURN;
0547 1 END;
0548 1
0549 1 ! Routine Init_log
```

```

                                .PSECT $SPLITS,NOWRT,NOEXE,2
47 4F 4C 24 47 49 46 4E 4F 43 49 4E 000A0 P.AAO: .ASCII \NICONFIG$LOG\
                                010E000C 000AC P.AAN: .LONG 17694732
                                00000000 000B0 .ADDRESS P.AAO
                                .EXTRN SYS$TRNLOG
                                .PSECT $CODE$,NOWRT,2
                                0000 00000 INIT_LOG:
                                SE      10 C2 00002 .WORD Save nothing
                                0000' CF D4 00005 .SUBL2 #16, SP
                                0A DD 00009 .CLRL CNF$GL_LOGMASK
                                04 AE 08 AE 9E 0000B .PUSHL #10
                                7E 7C 00010 .MOVAB RSLBFR, RSLDSC+4
                                7E D4 00012 .CLRQ -(SP)
                                0C AE 9F 00014 .CLRL -(SP)
                                10 AE 9F 00017 .PUSHAB RSLDSC
                                0000' CF 9F 0001A .PUSHAB RSLDSC
                                06 FB 0001E .PUSHAB P.AAN
                                50 D1 00025 .CALLS #6, SYS$TRNLOG
                                11 12 00028 .CMPL R0, #1
                                0000' CF 9F 0002A .BNEQ 1$
                                08 AE DD 0002E .PUSHAB CNF$GL_LOGMASK
                                08 AE DD 00031 .PUSHL RSLDSC+4
                                03 FB 00034 .PUSHL RSLDSC
                                04 0003B 1$: .CALLS #3, LIB$CVT_HTB
                                RET
                                : 0492
                                : 0522
                                : 0523
                                : 0524
                                : 0533
                                :
                                : 0535
                                : 0539
                                : 0541
                                : 0540
                                : 0545
; Routine Size: 60 bytes. Routine Base: $CODE$ + 00F0
```

```

365 0546 1 %SBTTL 'init_data      Initialize data structures'
366 0547 1 ROUTINE INIT_DATA : NOVALUE =
367 0548 1
368 0549 1 ++
369 0550 1
370 0551 1 This routine initializes the internal data structures.
371 0552 1
372 0553 1 --
373 0554 2 BEGIN
374 0555 2
375 0556 2 Initialize doubly linked list heads
376 0557 2
377 0558 2
378 0559 2
379 0560 2 List of circuits
380 0561 2
381 0562 2 CNF$GQ_CIRSURLST [0] = CNF$GQ_CIRSURLST [0];
382 0563 2 CNF$GQ_CIRSURLST [1] = CNF$GQ_CIRSURLST [0];
383 0564 2
384 0565 2
385 0566 2 List of Interrupt Request Blocks
386 0567 2
387 0568 2 CNF$GQ_IRBLST [0] = CNF$GQ_IRBLST [0];
388 0569 2 CNF$GQ_IRBLST [1] = CNF$GQ_IRBLST [0];
389 0570 2
390 0571 2 CNF$S_VM = 0;      ! For logging how much virtual memory has been allocated
391 0572 2 RETURN;
392 0573 1 END;      ! Routine Init_data
```

0004 00000 INIT_DATA:

	52	0000'	CF	9E	00002	.WORD	Save R2	: 0547
	62		62	9E	00007	MOVAB	CNF\$GQ_CIRSURLST, R2	: 0547
04	A2		62	9E	0000A	MOVAB	CNF\$GQ_CIRSURLST, CNF\$GQ_CIRSURLST	: 0562
08	A2	08	A2	9E	0000E	MOVAB	CNF\$GQ_CIRSURLST, CNF\$GQ_CIRSURLST+4	: 0563
0C	A2	08	A2	9E	00013	MOVAB	CNF\$GQ_IRBLST, CNF\$GQ_IRBLST	: 0568
		0000'	CF	D4	00018	MOVAB	CNF\$GQ_IRBLST, CNF\$GQ_IRBLST+4	: 0569
			04	0001C	CLRL	CNF\$S_VM		: 0571
					RET			: 0573

; Routine Size: 29 bytes, Routine Base: \$CODE\$ + 012C

```
0574 1 %SBTTL 'declare_objnam Declare object name to Network'
0575 1 ROUTINE DECLARE_OBJNAM : NOVALUE =
0576
0577 1 ++
0578
0579 1 This routine declares its object name, $NICONFIG, to the Network
0580 1
0581 1 --
0582
0583 BEGIN
0584 LOCAL
0585     IOSB :      BBLOCK [8],      ! IO status block
0586     NFB :      BBLOCK [5],      ! Network function block for DECLNAME
0587     NFB_DESC :  VECTOR [2],      ! Descriptor of NFB
0588     STATUS;
0589
0590 OWN
0591     OBJNAM_DESC : BBLOCK [DSCSC_S_BLN] ! Object name is $NICONFIG
0592     INITIAL (%CHARCOUNT('%$NICONFIG'),
0593             UPLIT PSECT ($OWNS) (%ASCII '$NICONFIG')));
0594
0595
0596 STATUS = LIB$ASN_WTH_MBX ( %ASCII '-NET:' ! Assign channel to NETACP
0597                          0,0, ! mailbox MAXMSG, BUFQUO (ignored)
0598                          CNF$W_NETCHAN, ! Channel to NETACP
0599                          CNF$W_MBXCHAN); ! Channel to mailbox
0600
0601 IF NOT .STATUS
0602 THEN
0603     BEGIN
0604         CNF$EXIT (.STATUS); ! There was an error assigning the channel
0605     END; ! No point in continueing
0606
0607 NFB [NFB$B_FCT] = NFB$C_DECLNAME; ! Set function to DECLARE NAME
0608 NFB [1,0,32,0] = 0; ! When declaring a name, must be zero
0609
0610 NFB_DESC [0] = 5; ! Set up descriptor for NFB, size is 5 bytes
0611 NFB_DESC [1] = NFB;
0612
0613 STATUS = $QIOW ( FUNC = IOS$ACPCONTROL, ! Request object name declaration to network
0614                CHAN = .CNF$W_NETCHAN, ! Use assigned channel
0615                EFN = CNF$C_SYNCH_EFN, ! Synchronous Event flag number
0616                IOSB = IOSB, ! IO status block
0617                P1 = NFB_DESC, ! Network function block
0618                P2 = OBJNAM_DESC); ! Object name being declared
0619
0620 IF .STATUS
0621 THEN
0622     STATUS = .IOSB [0,0,16,0]; ! successful submission
0623     ! pick up final status
0624
0625 IF .STATUS EQL SS$_BADPARAM ! If object already defined
0626 THEN
0627     BEGIN
0628         CNF$TRACE (DBG$C_TRACE, ! Report logic problem
0629                 $DESCRIPTOR('TRACE'), $DESCRIPTOR('Object already defined'));
0630     CNF$EXIT (SS$_NORMAL); ! Go away quietly
0631     END;
```

```

: 451      0631 2
: 452      0632 2
: 453      0633 2
: 454      0634 2
: 455      0635 2
: 456      0636 2
: 457      0637 2
: 458      0638 2
: 459      0639 2
: 460      0640 1

```

```

IF NOT .STATUS
THEN
    BEGIN
        SIGNAL (CNF$ NETASN, 0, .STATUS);
        CNF$EXIT (CNF$ NETASN);
    END;
RETURN;
END;

```

! Signal an error

! Routine Declare_objnam

```

                                .PSECT $PLITS$,NOWRT,NOEXE,2
                                00 00 00 3A 54 45 4E 5F 000B4 P.AAR: .ASCII \ NET:\<0><0><0>
                                010E0005 000BC P.AAQ: .LONG 17694725
                                00000000' 000C0 .ADDRESS P.AAR
                                45 43 41 52 54 000C4 P.AAT: .ASCII \TRACE\
                                000C9 .BLKB 3
                                00000005 000CC P.AAS: .LONG 5
                                00000000' 000D0 .ADDRESS P.AAT
20 79 64 61 65 72 6C 61 20 74 63 65 6A 62 4F 000D4 P.AAV: .ASCII \Object already defined\
64 65 6E 69 66 65 64 000E3
                                000EA .BLKB 2
                                00000016 000EC P.AAU: .LONG 22
                                00000000' 000F0 .ADDRESS P.AAV

```

```

                                .PSECT $OWNS$,NOEXE,2
                                00 00 00 47 49 46 4E 4F 43 49 4E 24 00024 P.AAP: .ASCII \ $NICONFIG\<0><0><0>
                                00000009 00030 OBJNAM_DESC:
                                00000000' 00034 .LONG 9
                                .ADDRESS P.AAP

```

.EXTRN SYSSQIOW

.PSECT \$CODE\$,NOWRT,2

```

001C 00000 DECLARE_OBJNAM:
54 0000V CF 9E 00002 .WORD Save R2,R3,R4
53 00000000G 8F D0 00007 MOVAB CNF$EXIT, R4
5E 0000' CF 9F 00011 MOVL #CNF$ NETASN, R3
0000' CF 9F 00015 SUBL2 #24, SP
0000' 7E 7C 00019 PUSHAB CNF$W_MBXCHAN
0000' CF 9F 00015 PUSHAB CNF$W_NETCHAN
0000' 7E 7C 00019 CLRQ -(SP)
0000' CF 9F 00018 PUSHAB P.AAQ
00000000G 00 05 FB 0001F CALLS #5, LIB$ASN_WTH_MBX
52 50 D0 00026 MOVL R0, STATUS
05 52 E8 00029 BLBS STATUS, 1$
52 DD 0002C PUSHL STATUS
08 64 01 FB 0002E CALLS #1, CNF$EXIT
AE 15 90 00031 1$: MOVAB #21, NFB
09 AE D4 00035 CLRL NFB+1
6E 05 D0 00038 MOVL #5, NFB_DESC
04 AE 08 AE 9E 0003B MOVAB NFB, NFB_DESC+4

```

0575

0596

0601

0604

0607

0608

0610

0611

		7E	7C	00040	CLRQ	-(SP)	0618
		7E	7C	00042	CLRQ	-(SP)	
	0000'	CF	9F	00044	PUSHAB	OBJNAM_DESC	
	14	AE	9F	00048	PUSHAB	NFB_DESC	
		7E	7C	0004B	CLRQ	-(SP)	
	30	AE	9F	0004D	PUSHAB	IOSB	
		38	DD	00050	PUSHL	#56	
	7E	0000'	CF	3C	MOVZWL	CNFSW_NETCHAN, -(SP)	
			01	DD	PUSHL	#1	
00000000G	00		0C	FB	CALLS	#12, SYS\$QIOW	
	52		50	DD	MOVL	R0, STATUS	
	04		52	E9	BLBC	STATUS, 2\$	0620
	52	10	AE	3C	MOVZWL	IOSB, STATUS	0622
	14		52	D1	CMPL	STATUS, #20	0624
			14	12	BNEQ	3\$	
		0000'	CF	9F	PUSHAB	P.AAU	0628
		0000'	CF	9F	PUSHAB	P.AAS	
			01	DD	PUSHL	#1	0627
0000V	CF		03	FB	CALLS	#3, CNF\$TRACE	
			01	DD	PUSHL	#1	0629
	64		01	FB	CALLS	#1, CNF\$EXIT	
	12		52	E8	BLBS	STATUS, 4\$	0632
			52	DD	PUSHL	STATUS	0635
			7E	D4	CLRL	-(SP)	
			53	DD	PUSHL	R3	
00000000G	00		03	FB	CALLS	#3, LIB\$SIGNAL	
			53	DD	PUSHL	R3	0636
	64		01	FB	CALLS	#1, CNF\$EXIT	
			04	00098	RET		0640

: Routine Size: 153 bytes, Routine Base: \$CODE\$ + 0149

```
462 0641 1 XSBTTL 'Set_time_bomb Wait for a set command before starting surveillance'
463 0642 1 ROUTINE SET_TIME_BOMB : NOVALUE =
464 0643 1
465 0644 1
466 0645 1
467 0646 1
468 0647 1
469 0648 1
470 0649 1
471 0650 1
472 0651 1
473 0652 1
474 0653 1
475 0654 1
476 0655 1
477 0656 1
478 0657 1
479 0658 1
480 0659 1
481 0660 1
482 0661 1
483 0662 1
484 0663 1
485 0664 1
486 0665 1
487 0666 1
488 0667 1
489 0668 1
490 P 0669 1
491 0670 1
492 0671 1
493 0672 1
494 P 0673 1
495 P 0674 1
496 0675 1
497 0676 1
498 0677 1
499 0678 1
500 0679 1
501 0680 1
502 0681 1
503 0682 1

--
This routine issues a read to the mailbox and waits for a SET
command from the initiator who began execution of NICONFIG.
If no command is forthcoming, NICONFIG quietly goes away.
--

BEGIN
LOCAL
STATUS;

Issue a wait and set an AST routine to go off.
That AST routine will queue a routine to the work queue that
will end the startup 'grace' period.
Then, if no surveillance requests have been received NICONFIG
will quietly disappear, otherwise it will remain until all
surveillance is turned off. This 'grace' period is to avoid multiple
false starts when someone does a show, realizes NICONFIG is not there
and then issues a set to start it up.

CNFSB_STARTING_UP = TRUE;

STATUS = $BINTIM ( TIMBUF = CNFSQ_A_STARTUP_WAIT, ! Convert ascii time to binary time
                  TIMADR = CNFSQ_B_STARTUP_WAIT);
IF NOT .STATUS THEN SIGNAL (CNFS_LOGIC, 0, .STATUS);

STATUS = $SETIMR ( EFN = CNFSC_STARTUP_EFN, ! Set the timer
                  DAYTIM = CNFSQ_B_STARTUP_WAIT,
                  ASTADR = TIME_BOMB); ! Routine to call when timer goes off
IF NOT .STATUS THEN SIGNAL (CNFS_LOGIC, 0, .STATUS);

CNF$TRACE (DBG$C TRACE, $DESCRIPTOR('TRACE'),
           $DESCRIPTOR('Startup verification set'));

RETURN;
END;

! Routine Set_time_bomb
```

```

.PSECT $PLITS,NOWRT,NOEXE,2
45 43 41 52 54 000F4 P.AAX: .ASCII \TRACE\
000F9 .BLKB 3
00000005 000FC P.AAW: .LONG 5
00000000 00100 .ADDRESS P.AAX
63 69 66 69 72 65 76 20 70 75 74 72 61 74 53 00104 P.AAZ: .ASCII \Startup verification set\
74 65 73 20 6E 6F 69 74 61 00113
00000018 0011C P.AAY: .LONG 24
00000000 00120 .ADDRESS P.AAZ

.EXTRN SYSSBINTIM, SYSSSETIMR
```

```
.PSECT $CODE$,NOWRT,2

001C 00000 SET_TIME_BOMB:
      54 00000000G 00 9E 00002      .WORD      Save R2,R3,R4      0642
      53 00000000G 8F D0 00009      MOVAB      LIB$SIGNAL, R4
      0000' CF      01 D0 00010      MOVL       #CNF$ LOGIC, R3
      0000'      CF 9F 00015      MOVL       #1, CNF$B_STARTING_UP      0667
      0000'      CF 9F 00019      PUSHAB     CNF$Q_B_STARTUP_WAIT      0670
      00000000G 00      02 FB 0001D      PUSHAB     CNF$Q_A_STARTUP_WAIT
      52      50 D0 00024      CALLS      #2, SYS$BINTIM
      09      52 E8 00027      MOVL       R0, STATUS
      52 DD 0002A      BLBS       STATUS, 1$
      7E D4 0002C      PUSHL      STATUS
      53 DD 0002E      CLRL       -(SP)
      64      03 FB 00030      PUSHL      R3
      0000V      7E D4 00033 1$:      CALLS      #3, LIB$SIGNAL
      0000'      CF 9F 00035      CLRL       -(SP)      0675
      0000'      CF 9F 00039      PUSHAB     TIME_BOMB
      00000000G 00      03 DD 0003D      PUSHAB     CNF$Q_B_STARTUP_WAIT
      52      04 FB 0003F      PUSHL      #3
      09      50 D0 00046      CALLS      #4, SYS$SETIMR
      52 E8 00049      MOVL       R0, STATUS
      52 DD 0004C      BLBS       STATUS, 2$
      7E D4 0004E      PUSHL      STATUS
      53 DD 00050      CLRL       -(SP)
      64      03 FB 00052      PUSHL      R3
      0000'      CF 9F 00055 2$:      CALLS      #3, LIB$SIGNAL
      0000'      CF 9F 00059      PUSHAB     P.AAY
      0000V      01 DD 0005D      PUSHAB     P.AAW
      CF      03 FB 0005F      PUSHL      #1
      0000V      03 FB 0005F      CALLS      #3, CNF$TRACE
      04 00064      RET
      04 00064
```

; Routine Size: 101 bytes, Routine Base: \$CODE\$ + 01E2

```

: 505      0683 1 XSBTTL 'time bomb Check whether startup should be aborted'
: 506      0684 1 ROUTINE TIME_BOMB : NOVALUE =
: 507      0685 1
: 508      0686 1
: 509      0687 1
: 510      0688 1
: 511      0689 1
: 512      0690 1
: 513      0691 1
: 514      0692 1
: 515      0693 2
: 516      0694 2
: 517      0695 2
: 518      0696 2
: 519      0697 2
: 520      0698 2
: 521      0699 2
: 522      0700 1

Queue routine to the work queue that will end
startup 'grace' period.

BEGIN
CNF$TRACE (DBG$C_TRACE, $DESCRIPTOR ('TRACE'),
$DESCRIPTOR ('Time_bomb --- End of grace period'));

WKQ$ADD_WORK_ITEM (TERMINATE_GRACE);      ! Terminate the startup period

RETURN TRUE;
END;                                     ! Routine Time_bomb

```

```

.PSECT $PLITS$,NOWRT,NOEXE,2

45 43 41 52 54 00124 P.ABB: .ASCII \TRACE\
00129
00000005 0012C P.ABA: .BLKB 3
00000000 00130 P.ABA: .LONG 5
45 20 2D 2D 2D 2D 62 6D 6F 62 5F 65 6D 69 54 00134 P.ABD: .ADDRESS P.ABB
72 65 70 20 65 63 61 72 67 20 66 6F 20 64 6E 00143 P.ABD: .ASCII \Time_bomb --- End of grace period\
64 6F 69 00152
00155
00000021 00158 P.ABC: .BLKB 3
00000000 0015C P.ABC: .LONG 33
ADDRESS P.ABD

```

```

.PSECT $CODE$,NOWRT,2

0000 00000 TIME_BOMB:
0000' CF 9F 00002 .WORD Save nothing
0000' CF 9F 00006 PUSHAB P.ABC
01 DD 0000A PUSHAB P.ABA
03 FB 0000C PUSHAB #1
0000V CF 0000V CF 9F 00011 CALLS #3, CNF$TRACE
0000G CF 01 FB 00015 PUSHAB TERMINATE GRACE
04 0001A CALLS #1, WKQ$ADD_WORK_ITEM
RET

```

; Routine Size: 27 bytes, Routine Base: \$CODE\$ + 0247

```

524 0701 1 XSBTTL 'terminate_grace Check whether startup should be aborted'
525 0702 1 ROUTINE TERMINATE_GRACE : NOVALUE =
526 0703 1
527 0704 1 ++
528 0705 1
529 0706 1 End startup 'grace' period. Now as soon as there are no longer any
530 0707 1 circuits under surveillance, NICONFIG will quietly go away
531 0708 1
532 0709 1 --
533 0710 2 BEGIN
534 0711 2 CNF$TRACE (DBG$C TRACE, $DESCRIPTOR ('TRACE'),
535 0712 2 $DESCRIPTOR ('Terminate_grace --- End of grace period'));
536 0713 2
537 0714 2 CNF$B_STARTING_UP = FALSE; ! Startup 'Grace' period is over
538 0715 2
539 0716 2 RETURN TRUE;
540 0717 1 END; ! Routine Terminate_grace

```

```

                                .PSECT $PLITS,NOWRT,NOEXE,2
                                45 43 41 52 54 00160 P.ABF: .ASCII \TRACE\
                                00165 .BLKB 3
                                00000005 00168 P.ABE: .LONG 5
                                00000000 0016C .ADDRESS P.ABF
65 63 61 72 67 5F 65 74 61 6E 69 6D 72 65 54 00170 P.ABH: .ASCII \Terminate_grace --- End of grace period\
61 72 67 20 66 6F 20 64 6E 45 20 2D 2D 2D 2D 0017F
                                0018E
                                00197 .BLKB 1
                                00000027 00198 P.ABG: .LONG 39
                                00000000 0019C .ADDRESS P.ABH

```

```

                                .PSECT $CODE$,NOWRT,2
                                0000 0000 TERMINATE GRACE:
                                .WORD Save nothing
                                0000' CF 9F 00002 PUSHAB P.ABG
                                0000' CF 9F 00006 PUSHAB P.ABE
                                0000V CF 01 DD 0000A PUSHL #1
                                0000' CF 03 FB 0000C CALLS #3, CNF$TRACE
                                0000' CF 04 00011 CLRL CNF$B_STARTING_UP
                                04 00015 RET
                                : 0702
                                : 0712
                                : 0711
                                :
                                : 0714
                                : 0717

```

; Routine Size: 22 bytes, Routine Base: \$CODE\$ + 0262

```
542 0718 1 ZSBTTL 'CNF$TRACE Log logic trace message to the Log'
543 0719 1 GLOBAL ROUTINE CNF$TRACE (LOGBITNUM, HEADDSC, TRACEDSC) : NOVALUE =
544 0720 1
545 0721 1 ++
546 0722 1 FUNCTIONAL DESCRIPTION:
547 0723 1
548 0724 1 Check the logging control mask and if the corresponding bit is set
549 0725 1 then print the special message to the log file. The message
550 0726 1 has a header and the tracing text.
551 0727 1
552 0728 1 FORMAL PARAMETERS:
553 0729 1
554 0730 1 logbitnum      Number of the logging bit to control the type of
555 0731 1                  logging
556 0732 1 headdsc        Address of a descriptor of the header text
557 0733 1 tracedsc       Address of a descriptor of the trace information
558 0734 1
559 0735 1 IMPLICIT INPUTS:
560 0736 1
561 0737 1 CNF$GL_LOGCONTROL
562 0738 1
563 0739 1 IMPLICIT OUTPUTS:
564 0740 1 NONE
565 0741 1
566 0742 1 ROUTINE VALUE:
567 0743 1 COMPLETION CODES:
568 0744 1 NONE
569 0745 1
570 0746 1 SIDE EFFECTS:
571 0747 1 NONE
572 0748 1
573 0749 1 --
574 0750 2 BEGIN
575 0751 2 BUILTIN
576 0752 2 NULLPARAMETER; ! Check if parameter was passed to routine
577 0753 2 MAP
578 0754 2 HEADDSC : REF BBLOCK,
579 0755 2 TRACEDSC : REF BBLOCK;
580 0756 2 LITERAL
581 0757 2 FAOSIZ = 256; ! The print buffer
582 0758 2 LOCAL
583 0759 2 FAOBUF : VECTOR [FAOSIZ, BYTE], ! Print buffer
584 0760 2 FAOLST : VECTOR [8, LONG], ! List of args to $FAOL
585 0761 2 OUTDSC : VECTOR [2]; ! Descriptor of the output line
586 0762 2
587 0763 2 !
588 0764 2 See if this text should be logged, and if not then return
589 0765 2
590 0766 2
591 0767 2 IF NOT .CNF$GL_LOGMASK [.LOGBITNUM]
592 0768 2 THEN
593 0769 2 RETURN;
594 0770 2
595 0771 2 OUTDSC [0] = FAOSIZ; ! Initialize the output buffer dsc
596 0772 2 OUTDSC [1] = FAOBUF;
597 0773 2 FAOLST [0] = .HEADDSC; ! Header text
598 0774 2 IF NULLPARAMETER (3)
```

```
599      0775      2      THEN
600      0776      FAOLST [1] = 0
601      0777      ELSE
602      0778      FAOLST [1] = .TRACEDSC;      ! Trace text dsc
603      0779      FAOLST [2] = 0;
604      0780      $FAOL
605      0781      ! Write the header out
606      0782      (
607      0783      CTRSTR = %ASCII '!/' !AS !AS!/' ,
608      0784      OUTLEN = OUTDSC [0],
609      0785      OUTBUF = OUTDSC,
610      0786      PRMLST = FAOLST
611      0787      );
612      0788
613      0789      LIB$PUT_OUTPUT (OUTDSC);
614      0790      RETURN;
615      0791      END;      ! Routine CNF$TRACE
```

```
00 2F 21 53 41 21 20 20 53 41 21 20 20 2F 21 001A0 P.ABJ: .ASCII \!/' !AS !AS!/'\<0>\<0>
00 00 001AF
010E000E 001B0 P.ABI: .LONG 17694734
00000000 001B4 .ADDRESS P.ABJ

.PSECT $SPLITS,NOWRT,NOEXE,2

.EXTRN SYSS$FAOL

.PSECT $CODE$,NOWRT,2

.ENTRY CNF$TRACE, Save nothing
MOVAB -296(SP), SP
BBC LOGBITNUM, CNF$GL_LOGMASK, 4$
MOVZWL #256, OUTDSC
MOVAB FAOBUF, OUTDSC+4
MOVL HEADDSC, FAOLST
CMPB (AP), #3
BLSSU 1$
TSTL 12(AP)
BNEQ 2$
CLRL FAOLST+4
BRB 3$
MOVL TRACEDSC, FAOLST+4
CLRL FAOLST+8
PUSHAB FAOLST
PUSHAB OUTDSC
PUSHAB OUTDSC
PUSHAB P.ABI
CALLS #4, SYSS$FAOL
PUSHL SP
CALLS #1, LIB$PUT_OUTPUT
RET
```

0719
0767
0771
0772
0773
0774
0776
0778
0779
0787
0789
0791

; Routine Size: 82 bytes, Routine Base: \$CODE\$ + 0278

```
617 0792 1 %SBTTL 'CNFSLOG_DATA Print a Data Message to the Log'
618 0793 1 GLOBAL ROUTINE CNFSLOG_DATA (LOGBITNUM, HEADDSC, EXTRADSC, DATADSC) : NOVALUE =
619 0794 1
620 0795 1 ++
621 0796 1 FUNCTIONAL DESCRIPTION:
622 0797 1
623 0798 1 Check the logging control mask and if the corresponding bit is set
624 0799 1 then print the special message to the log file. The message
625 0800 1 has a header and optionally some extra text which explains the
626 0801 1 logged message.
627 0802 1
628 0803 1 FORMAL PARAMETERS:
629 0804 1
630 0805 1     logbitnum      Number of the logging bit to control the type of
631 0806 1                   logging
632 0807 1
633 0808 1     headdsc        Address of a descriptor of the header text
634 0809 1
635 0810 1     extradsc       Address of a descriptor of the extra text (optional)
636 0811 1
637 0812 1     datadsc        Address of a descriptor of the data to be converted
638 0813 1                   and printed
639 0814 1
640 0815 1 IMPLICIT INPUTS:
641 0816 1
642 0817 1     CNF$GL_LOGCONTROL
643 0818 1
644 0819 1 IMPLICIT OUTPUTS:
645 0820 1     NONE
646 0821 1
647 0822 1 ROUTINE VALUE:
648 0823 1 COMPLETION CODES:
649 0824 1     NONE
650 0825 1
651 0826 1 SIDE EFFECTS:
652 0827 1     NONE
653 0828 1
654 0829 1 --
655 0830 1 BEGIN
656 0831 1 MAP
657 0832 1     HEADDSC      : REF BBLOCK,
658 0833 1     EXTRADSC     : REF BBLOCK,
659 0834 1     DATADSC      : REF BBLOCK;
660 0835 1 LITERAL
661 0836 1     FAOSIZ = 256;           ! The print buffer
662 0837 1 LOCAL
663 0838 1     FAOBUF : VECTOR [FAOSIZ, BYTE], ! Print buffer
664 0839 1     FAOLST : VECTOR [100],          ! List of args to $FAOL
665 0840 1     OUTDSC : VECTOR [2],            ! Descriptor of the output line
666 0841 1     BYTES,                          ! Counter for bytes written
667 0842 1     CTR : SIGNED,                  ! A random counter
668 0843 1     PTR,                            ! A random pointer
669 0844 1     ITR_CNT;                       ! Temporary iteration count
670 0845 1
671 0846 1
672 0847 1 See if data should be logged, and if not then return
673 0848 1
```

```
674 0849 2
675 0850
676 0851 IF NOT .CNF$GL_LOGMASK [.LOGBITNUM]
677 0852 THEN
678 0853 RETURN;
679 0854 OUTDSC [0] = FAOSIZ; ! Initialize the output buffer dsc
680 0855 OUTDSC [1] = FAOBUF;
681 0856 FAOLST [0] = .HEAD$DSC; ! Header text
682 0857 FAOLST [1] = .DATADSC [DSC$W_LENGTH]; ! Data length
683 0858 FAOLST [2] = ; ! Extra text dsc
684 0859 (
685 0860 IF .EXTRADSC EQL 0
686 0861 THEN
687 0862 %ASCID ''
688 0863 ELSE
689 0864 .EXTRADSC
690 0865 );
691 0866 $FAOL ! Write the header out
692 0867 (
693 0868 CTRSTR = %ASCID '!' / !AS (length = !UL bytes)! / !AS! / ',
694 0869 OUTLEN = OUTDSC [0],
695 0870 OUTBUF = OUTDSC,
696 0871 PRMLST = FAOLST
697 0872 );
698 0873 LIB$PUT_OUTPUT (OUTDSC);
699 0874
700 0875 CTR = .DATADSC [DSC$W_LENGTH]; ! Size of message
701 0876 PTR = .DATADSC [DSC$A_POINTER]; ! Its address
702 0877 WHILE .CTR GTR 0 DO ! Process it all
703 0878 BEGIN
704 0879 OUTDSC [0] = FAOSIZ; ! Set a descriptor
705 0880 OUTDSC [1] = FAOBUF;
706 0881 ITR CNT = MIN (.CTR, 20); ! Get byte count
707 0882 FAOLST [0] = .ITR CNT; ! Add count parameter
708 0883 FAOLST [.ITR CNT+1] = .ITR CNT;
709 0884 FAOLST [(.ITR CNT+1)*2] = .ITR CNT;
710 0885 INCRU IDX FROM 1 TO .ITR_CNT DO ! A few bytes at a time
711 0886 BEGIN
712 0887 FAOLST [.IDX] = (.PTR) <0, 8, 0>; ! One for the hex
713 0888 FAOLST [.IDX + .ITR_CNT+1] = (.PTR) <0, 8, 0>; ! Decimal
714 0889 FAOLST [2*(.IDX + .ITR_CNT)+1] = 1; ! One for character
715 0890 FAOLST [2*(.IDX + .ITR_CNT)+1 + 1] = .PTR;
716 0891 PTR = .PTR + 1; ! Next one
717 0892 CTR = .CTR - 1; ! One less
718 0893 END;
719 0894
720 0895 $FAOL ! Saviour of bored programmers
721 0896 (
722 0897 CTRSTR = %ASCID '!' # (4XB)! / ! # (4UB)! / ! # (4AF)! / ',
723 0898 OUTLEN = OUTDSC [0],
724 0899 OUTBUF = OUTDSC,
725 0900 PRMLST = FAOLST
726 0901 );
727 0902
728 0903 LIB$PUT_OUTPUT (OUTDSC); ! Write to SYS$OUTPUT
729 0904 END;
730 0905 ! CNF$LOG_DATA
```

```
74 67 6E 65 6C 28 20 20 53 41 21 20 20 2F 21 001B8 P.ABL: .BLKB 0
21 29 73 65 74 79 62 20 4C 55 21 20 3D 20 68 001B8 P.ABK: .LONG 17694720
                                00000000' 001BC .ADDRESS P.ABL
                                001C0 P.ABN: .ASCII \!/\ !AS (length = !UL bytes)!/ !AS!/-
                                001CF \<0>
                                001DE
                                001E7
                                010E0026' 001E8 P.ABM: .ASCII <0>
                                00000000' 001EC .LONG 17694758
                                001F0 P.ABP: .ADDRESS P.ABN
                                001FF P.ABP: .ASCII \!#(4XB)!/!#(4UB)!/ !#(4AF)!/\<0><0>
                                0020E
                                010E001E' 00210 P.ABO: .LONG 17694750
                                00000000' 00214 .ADDRESS P.ABP
```

```
01 0000' 57 00000000G 00 00FC 00000 .ENTRY CNF$LOG_DATA, Save R2,R3,R4,R5,R6,R7 0793
56 00000000G 00 9E 00002 MOVAB LIB$PUT_OUTPUT, R7
5E FD68 CE 9E 00010 MOVAB SYS$FAOL, R6
CF 04 AC E0 00015 MOVAB -664(SP), SP
                                04 0001C BBS LOGBITNUM, CNF$GL_LOGMASK, 1$ 0850
                                04 0001C RET
                                04 0001D 1$: MOVZWL #256, OUTDSC 0854
04 AE FF00 CD 9E 00022 MOVAB FAOBUF, OUTDSC+4 0855
08 AE 08 AC D0 00028 MOVL HEADDSC, FAOLST 0856
52 10 AC D0 0002D MOVL DATADSC, R2 0857
OC AE 62 3C 00031 MOVZWL (R2), FAOLST+4
                                0C AC D5 00035 TSTL EXTRADSC 0860
                                07 12 00038 BNEQ 2$
50 0000' CF 9E 0003A MOVAB P.ABK, R0 0861
                                04 11 0003F BRB 3$
10 50 0C AC D0 00041 2$: MOVL EXTRADSC, R0 0864
                                50 D0 00045 3$: MOVL R0, FAOLST+8 0859
                                08 AE 9F 00049 PUSHAB FAOLST 0872
                                04 AE 9F 0004C PUSHAB OUTDSC
                                08 AE 9F 0004F PUSHAB OUTDSC
                                0000' CF 9F 00052 PUSHAB P.ABM
66 04 FB 00056 CALLS #4, SYS$FAOL 0873
5E DD 00059 PUSHL SP
67 01 FB 0005B CALLS #1, LIB$PUT_OUTPUT 0875
53 62 3C 0005E MOVZWL (R2), CTR 0876
55 04 A2 D0 00061 MOVL 4(R2), PTR 0877
                                53 D5 00065 4$: TSTL CTR
                                72 15 00067 BLEQ 8$
04 6E 0100 8F 3C 00069 MOVZWL #256, OUTDSC 0879
AE FF00 CD 9E 0006E MOVAB FAOBUF, OUTDSC+4 0880
50 53 D0 00074 MOVL CTR, R0
14 50 D1 00077 CML R0, #20
03 15 0007A BLEQ 5$ 0881
```

	50		14	D0	0007C	MOVL	#20, R0		
	52		50	D0	0007F	5\$:	MOVL	R0, ITR_CNT	
	08 AE		52	D0	00082		MOVL	ITR_CNT, FAOLST	0882
	0C AE42		52	D0	00086		MOVL	ITR_CNT, FAOLST+4[ITR_CNT]	0883
50	52		01	78	0008B		ASHL	#1, ITR_CNT, R0	0884
	10 AE40		52	D0	0008F		MOVL	ITR_CNT, FAOLST+8[R0]	
	51		01	D0	00094		MOVL	#1, -IDX	0885
			26	11	00097		BRB	7\$	
	08 AE41		65	9A	00099	6\$:	MOVZBL	(PTR), FAOLST[IDX]	0887
50	51		52	C1	0009E		ADDL3	ITR_CNT, IDX, R0	0888
	0C AE40		65	9A	000A2		MOVZBL	(PTR), FAOLST+4[R0]	
	54	8142	9E	000A7			MOVAB	(IDX)+[ITR_CNT], R4	0889
50	54		01	78	000AB		ASHL	#1, R4, R0	
	0C AE40		01	D0	000AF		MOVL	#1, FAOLST+4[R0]	
50	54		01	78	000B4		ASHL	#1, R4, R0	0890
	10 AE40		85	9E	000B8		MOVAB	(PTR)+, FAOLST+8[R0]	
			53	D7	000BD		DECL	CTR	0892
	52		51	D1	000BF	7\$:	CMPL	IDX, ITR_CNT	0885
			D5	1B	000C2		BLEQU	6\$	
		08	AE	9F	000C4		PUSHAB	FAOLST	0901
		04	AE	9F	000C7		PUSHAB	OUTDSC	
		08	AE	9F	000CA		PUSHAB	OUTDSC	
		0000	CF	9F	000CD		PUSHAB	P.ABO	
	66		04	FB	000D1		CALLS	#4, SYSS\$FAOL	
			5E	DD	000D4		PUSHL	SP	0903
	67		01	FB	000D6		CALLS	#1, LIB\$PUT_OUTPUT	
			8A	11	000D9		BRB	4\$	0877
			04	000DB	8\$:		RET		0905

; Routine Size: 220 bytes, Routine Base: \$CODE\$ + 02CA

```

732 0906 1 %SBTTL 'CNF$EXIT Clean up and exit'
733 0907 1 GLOBAL ROUTINE CNF$EXIT (STATUS) : NOVALUE =
734 0908 1
735 0909 1 ++
736 0910 1 FUNCTIONAL DESCRIPTION:
737 0911 1
738 0912 1 Permit a graceful exit for $NICONFIG
739 0913 1
740 0914 1 FORMAL PARAMETERS:
741 0915 1
742 0916 1 Status Code to exit with.
743 0917 1
744 0918 1 IMPLICIT INPUTS:
745 0919 1
746 0920 1 IMPLICIT OUTPUTS:
747 0921 1 NONE
748 0922 1
749 0923 1 SIDE EFFECTS:
750 0924 1
751 0925 1 Terminate program execution
752 0926 1
753 0927 1 --
754 0928 1
755 0929 2 BEGIN
756 0930 2 CNF$TRACE (DBG$C TRACE, $DESCRIPTOR('TRACE'),
757 0931 2 $DESCRIPTOR('$EXIT'));
758 0932 2 $EXIT (CODE= .STATUS);
759 0933 1 END; ! Routine EXIT

```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
45 43 41 52 54 00218 P.ABR: .ASCII \TRACE\
0021D .BLKB 3
00000005 00220 P.ABQ: .LONG 5
00000000' 00224 .ADDRESS P.ABR
54 49 58 45 24 00228 P.ABT: .ASCII \EXIT\
0022D .BLKB 3
00000005 00230 P.ABS: .LONG 5
00000000' 00234 .ADDRESS P.ABT

.EXTRN SYS$EXIT

.PSECT $CODE$,NOWRT,2

0000' 0000 0000
0000' CF 9F 00002
0000' CF 9F 00006
FEC1 CF 01 DD 0000A
00000000G 00 04 AC DD 00011
01 FB 00014
04 0001B

.ENTRY CNF$EXIT, Save nothing
PUSHAB P.ABS
PUSHAB P.ABQ
PUSHL #1
CALLS #3, CNF$TRACE
PUSHL STATUS
CALLS #1, SYS$EXIT
RET

```

; Routine Size: 28 bytes, Routine Base: \$CODE\$ + 03A6

CNFMAIN
V04-000

DECnet Ethernet Configurator Module
CNFSEXIT Clean up and exit

H 13
16-Sep-1984 02:02:49
14-Sep-1984 12:49:51

VAX-11 Bliss-32 V4.0-742
[NICNF.SRC]CNFMAIN.B32;1

Page 29
(13)

```
761 0934 1 XSBTTL 'CNF$GET_ZVM Get zeroed virtual memory'
762 0935 1 GLOBAL ROUTINE CNF$GET_ZVM (SIZ_ADR, ADR) =
763 0936 1
764 0937 1 ++
765 0938 1 FUNCTIONAL DESCRIPTION:
766 0939 1
767 0940 1 This routine allocates virtual memory and zeros it.
768 0941 1 It provides a common point for reporting memory errors
769 0942 1 and logging memory usage.
770 0943 1
771 0944 1 FORMAL PARAMETERS:
772 0945 1
773 0946 1 siz_adr Longword containing the number of bytes to allocate
774 0947 1
775 0948 1 adr Address of longword in which to return the starting
776 0949 1 address of the allocated memory.
777 0950 1
778 0951 1 IMPLICIT INPUTS:
779 0952 1
780 0953 1 CNF$GL_LOGMASK Determine if memory usage should be logged
781 0954 1 CNF$S_VM Record a running tally of total memory allocated
782 0955 1
783 0956 1 IMPLICIT OUTPUTS:
784 0957 1 NONE
785 0958 1
786 0959 1 ROUTINE VALUE:
787 0960 1 COMPLETION CODES:
788 0961 1 NONE
789 0962 1
790 0963 1 SIDE EFFECTS:
791 0964 1 NONE
792 0965 1
793 0966 1 --
794 0967 1
795 0968 1 BEGIN
796 0969 1 LOCAL
797 0970 1 STATUS;
798 0971 1
799 0972 1
800 0973 1 STATUS = LIB$GET_VM (.SIZ_ADR, .ADR); ! Get the memory
801 0974 1 IF NOT .STATUS
802 0975 1 THEN
803 0976 1 BEGIN
804 0977 1 SIGNAL_STOP (CNF$_FAILGETVM, 1, ..SIZ_ADR, .STATUS); ! Signal the error
805 0978 1 END;
806 0979 1
807 0980 1 IF .CNF$GL_LOGMASK [DBG$C_VM] ! If memory logging is enabled
808 0981 1 THEN
809 0982 1 BEGIN
810 0983 1 CNF$S_VM = .CNF$S_VM + ..SIZ_ADR; ! Tally it,
811 0984 1 SIGNAL (CNF$_GETVM, 2, ..SIZ_ADR, .CNF$S_VM); ! and report it.
812 0985 1 END;
813 0986 1
814 0987 1 CH$FILL (0, ..SIZ_ADR, ..ADR); ! Zero it
815 0988 1 RETURN TRUE;
816 0989 1 END; ! Routine CNF$GET_ZVM
```

				003C 00000	.ENTRY CNF\$GET_ZVM, Save R2,R3,R4,R5	0935
				AC 7D 00002	MOVQ SIZ_ADR, -(SP)	0973
	000000C0G	7E 04		02 FB 00006	CALLS #2, LIB\$GET_VM	
		14		50 E8 0000D	BLBS STATUS, 1\$	0974
				50 DD 00010	PUSHL STATUS	0977
			04	BC DD 00012	PUSHL @SIZ_ADR	
				01 DD 00015	PUSHL #1	
				8F DD 00017	PUSHL #CNF\$_FAILGETVM	
	00000000G	00 00000000G		04 FB 0001D	CALLS #4, LIB\$STOP	
1C	0000'	CF		02 E1 00024	BBC #2, CNF\$GL_LOGMASK, 2\$	0980
	0000'	CF		BC C0 0002A	ADDL2 @SIZ_ADR, CNF\$L_VM	0983
			04	CF DD 00030	PUSHL CNF\$C_VM	0984
			04	BC DD 00034	PUSHL @SIZ_ADR	
				02 DD 00037	PUSHL #2	
				8F DD 00039	PUSHL #CNF\$_GETVM	
	00000000G	00 00000000G		04 FB 0003F	CALLS #4, LIB\$SIGNAL	
			08	BC D0 00046	MOVL @ADR, R0	0987
04	BC	00		00 2C 0004A	MOVCS #0, (SP), #0, @SIZ_ADR, (R0)	
				60 00050		
				01 D0 00051	MOVL #1, R0	0988
				04 00054	RET	0989

; Routine Size: 85 bytes, Routine Base: \$CODE\$ + 03C2

```

818 0990 1 XSBTTL 'CNF$FREE_VM Free virtual memory'
819 0991 1 GLOBAL ROUTINE CNF$FREE_VM (SIZ_ADR, ADR) =
820 0992 1
821 0993 1 ++
822 0994 1 FUNCTIONAL DESCRIPTION:
823 0995 1
824 0996 1 This routine deallocates virtual memory.
825 0997 1 It provides a common point for reporting memory errors
826 0998 1 and logging memory usage.
827 0999 1
828 1000 1 FORMAL PARAMETERS:
829 1001 1
830 1002 1 siz_adr Longword containing the number of bytes to deallocate
831 1003 1
832 1004 1 adr Address of longword in containing the starting
833 1005 1 address of the allocated memory.
834 1006 1
835 1007 1 IMPLICIT INPUTS:
836 1008 1
837 1009 1 CNF$GL_LOGMASK Determine if memory usage should be logged
838 1010 1 CNF$S_VM Record a running tally of total memory allocated
839 1011 1
840 1012 1 IMPLICIT OUTPUTS:
841 1013 1 NONE
842 1014 1
843 1015 1 ROUTINE VALUE:
844 1016 1 COMPLETION CODES:
845 1017 1
846 1018 1 NONE
847 1019 1
848 1020 1 SIDE EFFECTS:
849 1021 1
850 1022 1 NONE
851 1023 1
852 1024 1 --
853 1025 1
854 1026 2 BEGIN
855 1027 2 LOCAL
856 1028 2 STATUS;
857 1029 2
858 1030 2 STATUS = LIB$FREE_VM (.SIZ_ADR, .ADR); ! Deallocate it
859 1031 2 IF NOT .STATUS
860 1032 2 THEN
861 1033 2 BEGIN ! Report any errors
862 1034 2 SIGNAL (CNF$_FAILFREV, 1, ..SIZ_ADR, .STATUS);
863 1035 2 .ADR = 0;
864 1036 2 RETURN .STATUS;
865 1037 2 END;
866 1038 2
867 1039 2 IF .CNF$GL_LOGMASK [DBG$C_VM] ! If memory logging is enabled
868 1040 2 THEN
869 1041 2 BEGIN
870 1042 2 CNF$S_VM = .CNF$S_VM - ..SIZ_ADR; ! Update tally
871 1043 2 SIGNAL (CNF$_FREEVM, 2, ..SIZ_ADR, .CNF$S_VM); ! and report it.
872 1044 2 END;
873 1045 2
874 1046 2 RETURN TRUE;
```

; 875

1047 1 END;

! Routine CNF\$FREE_VM

				000C 00000	.ENTRY	CNF\$FREE_VM, Save R2,R3		0991	
		53	00000000G	00	9E	00002	MOVAB	LIB\$SIGNAL, R3	
		7E	04	AC	7D	00009	MOVQ	SIZ_ADR, -(SP)	1030
	00000000G	00		02	FB	0000D	CALLS	#2, LIB\$FREE_VM	
		52		50	D0	00014	MOVL	R0, STATUS	
		17		52	E8	00017	BLBS	STATUS, 1\$	1031
				52	DD	0001A	PUSHL	STATUS	1034
			04	BC	DD	0001C	PUSHL	@SIZ_ADR	
				01	DD	0001F	PUSHL	#1	
			00000000G	8F	DD	00021	PUSHL	#CNF\$ FAILFREV	
		63		04	FB	00027	CALLS	#4, LIB\$SIGNAL	
			08	BC	D4	0002A	CLRL	@ADR	1035
		50		52	D0	0002D	MOVL	STATUS, R0	1036
				04	00030		RET		
18	0000'	CF		02	E1	00031	BBC	#2, CNF\$GL_LOGMASK, 2\$	1039
	0000'	CF		BC	C2	00037	SUBL2	@SIZ_ADR, CNF\$SL_VM	1042
			04	CF	DD	0003D	PUSHL	CNF\$C_VM	1043
			04	BC	DD	00041	PUSHL	@SIZ_ADR	
				02	DD	00044	PUSHL	#2	
			00000000G	8F	DD	00046	PUSHL	#CNF\$ FREEVM	
		63		04	FB	0004C	CALLS	#4, LIB\$SIGNAL	
		50		01	D0	0004F	MOVL	#1, R0	1046
				04	00052		RET		1047

; Routine Size: 83 bytes, Routine Base: \$CODE\$ + 0417

CNFMMAIN
V04-000

DECnet Ethernet Configurator Module
CNFSFREE_VM Free virtual memory

M 13
16-Sep-1984 02:02:49
14-Sep-1984 12:49:51

VAX-11 Bliss-32 V4.0-742
[NICNF.SRC]CNFMMAIN.B32;1

Page 34
(16)

: 877
: 878
: 1048 1 END
: 1049 0 ELUDOM

!End of module CNFMMAIN

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	156	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	56	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	568	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	1130	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
. ABS	0	NOVEC, NOWRT, NORD, NOEXE, NOSHR, LCL, ABS, CON, NOPIC, ALIGN(0)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	21	0	581	00:01.1
-\$255\$DUA28:[SHRLIB]NET.L32;1	1279	2	0	63	00:00.9

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:CNFMMAIN/OBJ=OBJ\$:CNFMMAIN MSRC\$:CNFMMAIN/UPDATE=(ENH\$:CNFMMAIN)

: Size: 1130 code + 780 data bytes
: Run Time: 00:22.8
: Elapsed Time: 00:42.5
: Lines/CPU Min: 2766
: Lexemes/CPU-Min: 20225
: Memory Used: 130 pages
: Compilation Complete

0279 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY